Real World Application of Event-based End to End Autonomous Driving

Yuxuan Chen
MIT PRIMES
Mentored by Igor Gilitshenski and Alexander Amini

Abstract

End-to-end autonomous driving has recently been a popular area of study for deep learning. This work studies the use of event cameras for real-world deep learned driving task in comparison to traditions RGB cameras. In this work, we evaluate existing state-of-the-art event-based models on offline datasets, design a novel model that fuses the benefits from both event-based and traditional frame-based cameras, and integrate the trained models on board a full-scale vehicle. We conduct tests in a challenging track with features unseen to the model. Through our experiments and saliency visualization, we show that event-based models actually predict the existing motion of the car rather than the active control the car should take. Therefore, while event-based models excel at offline tasks such as motion estimation, our experiments reveal a fundamental challenge in applying event-based end-to-end learning to active control tasks, that the models need to learn reasoning about future actions with a feedback loop that impacts its future state.

1 Introduction

Autonomous vehicles need the ability to quickly react to sudden changes, deal with constantly changing illumination, and be robust driving at a high speed. However, traditional frame-based cameras have high latency, perform poorly under insufficient illumination or sun glare, and suffer from motion blur under rapid movements.

Event-based cameras, such as the Dynamic Vision Sensor (DVS) [1], are bio-inspired sensors that work radically different from traditional cameras. Instead of outputting frames with a fixed size at a fixed rate, event cameras have pixels that independently output asynchronous brightness changes (called "events"). Therefore, the output of an event-based camera is a stream of asynchronous events, each described by its time, location, and sign of brightness change. Due to this special design, event cameras have many advantages over traditional cameras: low latency (μs vs. ms), high dynamic range (140 dB vs. 60dB), and no motion blur [2], making them a perfect fit for autonomous driving over traditional frame-based cameras. Moreover, event cameras are natural motion detectors and have shown great potential in traditional computer vision tasks such as object classification and localization [3, 4, 5].

Recently, deep learning has been proposed as a novel solution to traditional computer vision tasks for both event cameras and traditional RGB cameras. In this work, we develop a novel end-to-end model that fuses the benefits from both event-based and classical vision sensors to learn a full-scale autonomous vehicle controller. We demonstrate feasibility of our learned model through deployment onto real-world roads and evaluate against competing state-of-the-art approaches. To the best of our knowledge, we are the first to actuate a full-scale autonomous vehicle end-to-end with the help of an event camera.

Overall, our key contributions are as follows:

1. Design of a novel end-to-end driving model which processes both event-based and traditional RGB-based input modalities;

2. Evaluation of our model on offline driving datasets, demonstrating the strength of event-based models;

3. Interpretation of our model through saliency visualization and ablation study; and

4. Integration and deployment in a full-scale autonomous vehicle, demonstrating the challenges for real-world end-to-end driving using an event-based camera.

The remainder of this paper is organized as follows: Sec. 2 reviews related work on the problem. Sec. 3 formulates our models. Sec. 4 introduces the experiment setup. Sec. 5 describes our datasets and experiment setup. And Sec. 6 draws the conclusion.

2 Related Work

While traditional approaches to autonomous driving divide the problem into several sub-problems like mapping and localization [6, 7], perception [8, 9], planning [10, 11], and control [12, 13], each solved with specific algorithms, end-to-end learning attempts to use deep neural networks to map raw sensor input directly to control. This idea of end-to-end learning was first proposed in 1989 in the ALVINN system [14], where a shallow fully connected network showed the ability to steer a car on public roads. More recently, NVIDIA demonstrated the ability for a CNN, named PilotNet, to learn driving policy directly from video frames [15]. PilotNet's implementation was tested on a real car and, therefore, is an important foundation for our work.

There have been numerous extensions of end-to-end driving with additional supported capabilities such as navigating through intersections, modeling a probabilistic action space, and performing lane-changes [16, 17]. All of these techniques for end-to-end learning rely on traditional frame-based cameras and thus are susceptible to high latency, low dynamic range, and motion blur [2]. In contrast, we introduce the use of event-based cameras in deployable end-to-end driving.

Event cameras have shown success in the field of robotics with the help of deep learning. In [5], the authors proposed a deep-learning method to steer a predator robot towards another prey robot. They adopted a CNN structure that takes input alternatively from both the traditional camera and integrated event-frames and classify the location of the pray robot to 4 classes: left, middle, right, or not present. However, our work differs in that our network takes input from the traditional camera and integrated frames concurrently rather than alternatively, and our model trains end-to-end, outputting a continuous steering wheel angle that controls the car directly.

End-to-end event-based driving was first studied in [18], in which the authors demonstrated the ability for the model to perform well on a large and complex dataset. While the main goal of their work was to understand how learning-based approaches to motion-estimation task could benefit from the natural response of event cameras to motion, the model's ability to actually steer a vehicle in the real-world was not studied. In this work, we demonstrate that these existing approaches do not apply to an active testing scenario and instead design a novel model more capable of full-scale vehicle control.

3 Our Models

We first implemented two existing models to be tested on our autonomous vehicle as the baseline: PilotNet [15], taking input from a traditional RGB cameras, and the model proposed by Maqueda et al. [18], taking input from a event-based camera. While PilotNet has been tested in an autonomous vehicle, the results of [18] have only been on the dataset, so this implementation on the car is one of our contribution.

Through our initial experiments with [18], we find the event-camera to be sometimes unable to fully capture a scene, for example, when the car is moving slowly and there are few number of events. Therefore, we aim at augmenting the model with inputs from a traditional RGB camera. This is done through feeding the event-frames and the RGB-frames to two similar sequences of 5 convolutional layers, and then concatenating the outputs as the global features. The global features are then fed into 3 final fully connected layers leading to an output value corresponding to the curvature, which steers the car directly.

As our model fuses input from both the event-based camera and the RGB camera, we expect our model to perform no worse than the first two models and, potentially, outperform both models by combining the strengths of the two cameras. That is, we hope the model to focus on RGB data when meaningful events are insufficient, and, when possible, use events to achieve low latency, high dynamic range, and no motion blur.

We process raw events into event frames similar to that of [18]. However, to discourage the model from learning the current motion of the car revealed by event polarity, we integrate all events into a single channel. That is, given a time interval T and events $e_k = (x_k, y_k, t_k, p_k)$, [1] the resulting event frame histogram is:

$$H(x, y) = \sum_{t_k \in T} \delta(x - x_k, y - y_k), \tag{1}$$

[1]An event e_k represents a brightness change of a certain magnitude at spatial coordinate (x_k, y_k), time t_k, with the sign temporal information t_k, and polarity (sign of the brightness change) $p_k \in \{+1, -1\}$

where δ is Kronecker delta funciton. And to force the model to learn from features relevant to the driving task, such as lane markings and other vehicles, we crop the event frames to the center of the road, just like that of [15]. Moreover, instead of rescaling the images to [0, 1] before feeding into the model, we cap the each event count at one to lower the influence of event count on the model, making our event frames essentially binary images.

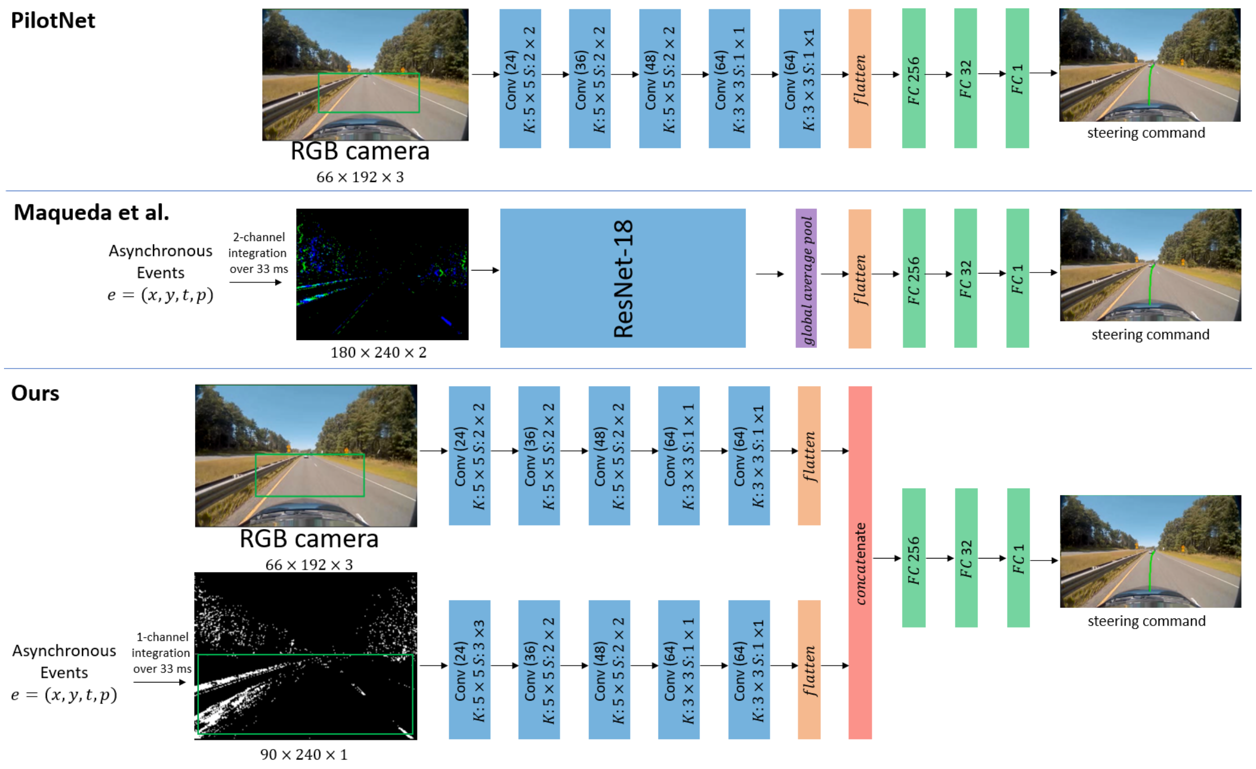

PilotNet
RGB camera
66 × 192 × 3
Conv (24) K: 5 × 5 S: 2 × 2
Conv (36) K: 5 × 5 S: 2 × 2
Conv (48) K: 5 × 5 S: 2 × 2
Conv (64) K: 3 × 3 S: 1 × 1
Conv (64) K: 3 × 3 S: 1 × 1
flatten
FC 256
FC 32
FC 1
steering command
Maqueda et al.
Asynchronous Events
e = (x, y, t, p)
2-channel integration over 33 ms
180 × 240 × 2
ResNet-18
global average pool
Ours
1-channel integration over 33 ms
90 × 240 × 1
Conv (24) K: 5 × 5 S: 3 × 3
concatenate

Figure 1: **Our implementation of PilotNet, model proposed by Maqueda et al.[18], and our model.**
Top: PilotNet uses a simple 5-layer CNN as the feature extractor.
Middle: Maqueda et al. integrates asynchronous events into a double-channel event frame and use a ResNet-18 as the feature extractor.
Bottom: Our model integrates asynchronous events into single-channel event frames. Two different PilotNet-style CNNs are used to extract features from both event-frames and RGB-frames, with the output tensors concatenated as the global features.
The green box shows the actual input to the model. ReLu [19] is used as the activation function. Batchnorm [20] is used for all layers except the output layer. Dropout layers [21] are used before the last three fully connected layers.

4 Experiment Setup

4.1 System Setup

We collected our datasets and evaluated our models on a 2015 Toyota Prius V equipped with autonomous drive-by-wire capabilities [22]. Similar to [23], we made several additions to the sensor and compute platform for this work. One forward-facing Leopard Imaging LI-AR0231-GMSL camera [24] is used as the RGB vision source for this study. Additionally, a forward-facing

DAVIS240 [25] sensor is used as the event-based vision source, as shown in Figure 2. We use the yaw rate λ $[rad/sec]$ and the speed of the vehicle v $[m/sec]$ to compute the curvature (or inverse steering radius) of the path which the human executed as $\theta_s = \frac{\lambda}{v}$. Finally, all of the sensor processing and computation was done on board an NVIDIA Drive PX2 [26].

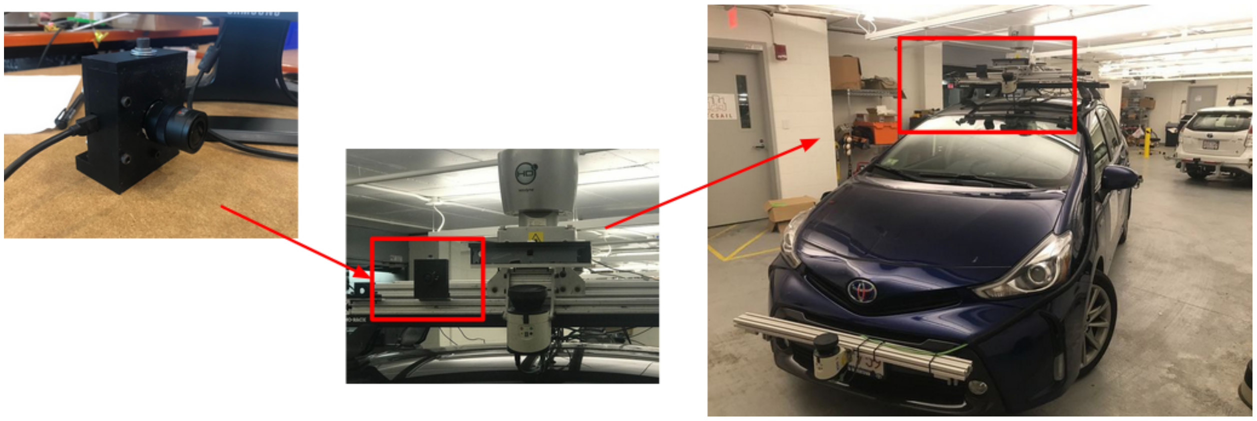


Figure 2: **DAVIS240 sensor position on the vehicle**

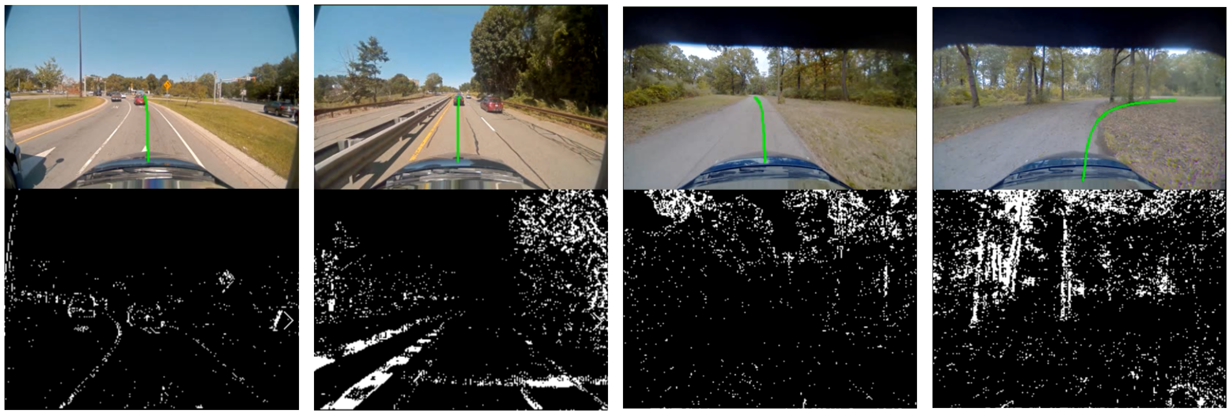
4.2 Data Collection and Processing

We recorded two separate one-hour urban driving datasets around the Boston area, one for training and the other for validation. We preprocessed the data similar to [15], which has been tested on a full-scale car. To train the model for simple lane-following, we only selected data where the driver was staying in a lane and discarded the rest. And as the data recorded has a strong bias for driving straight, we sample the training data to include a higher proportion of driving scenarios on curved roads.

4.3 Optimization

In our training process, we used the L1 norm between the predicted curvatures and curvatures executed by human driver as the loss function according to [27], for a better performance on the vehicle. We used a batch size of 32 and trained the models for 20 epochs with the Adam optimizer [28] with $\alpha = 10^{-5}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$.

For testing on a full-scale autonomous vehicle at our test site in Devens, MA, which contains rural roads without proper lane marking, we fine tuned our models trained on the urban dataset on another two minutes of Devens training data for 3 epochs, and validated again on another one minute of Devens validation dataset.



(a) urban 1 (b) urban 2 (c) Devens 1 (d) Devens 2

Figure 3: **Our datasets.**
The urban dataset contains roads with propar land markings, but the Devens dataset, representing our road-test location in Devens, MA, lacks propar land markings and contain extensive shadows and vegetation, making autonomous driving extremely challenging.
The images above show the RGB frames and the images below show the integrated event frames using our method. Both frames are not cropped to ROI for display purposes. The green curve shows the curvature executed by the human driver.

5 Experiments

5.1 Dataset Results

To measure model performance on the datasets, we adopt the same metrics as [18]. Given ground truths a and model predictions $\hat{a}$

$$RMSE = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(\hat{a}_j - a_j)^2} \tag{2}$$

$$EVA = 1 - \frac{Var(\hat{a} - a)}{Var(a)} \tag{3}$$

The root-mean-squared error (RMSE) measures the average magnitude of the prediction error, indicating how close the observed values are to those predicted by the network. The explained variance (EVA) measures the proportion of variation in the predicted values with respect to that of the observed values. If predicted values approximate the ground truth well, $Var(\hat{a} - a)$ would be smaller than $Var(a)$, resulting in an EVA>0.

With these metrics, we first evaluated the models stated in Section 3. Moreover, we also hope to understand the influence of our methods of Region Of Interest (ROI) cropping and 1-channel integration on dataset metric performance. Therefore, we performed ablation studies with models of [18], adding ROI-cropping and/or replacing 2-channel integration with 1-channel integration. The results are shown in Table 1.

In terms of RMSE results, we note that our model performs comparable to [18] and [15], as a curvature error of about 0.0003 corresponds to a steering wheel angle error of less than 2 degrees. In terms of EVA, a slight performance decrease from [18] in our model is to be expected, as our model is much simpler than the ResNet-18 used by [18] and our input discards much information that we deem unnecessary for steering angle prediction. However, it is obvious that event-based models perform generally better than frame-based models like PilotNet.

In the ablations, we notice that ROI cropping has a significant impact on the performance of [18], much more than 1-channel integration. With ROI cropping, the performance of [18] degrades to the level of PilotNet and is worse than our model. As PilotNet has shown the importance of ROI cropping in the model being able to drive an actual car, and we conjecture that two-channel integration may encourage the model to cheat by predicting the existing motion of the car, we wish to better understand the models through visualization.

Model	urban RMSE	urban EVA	Devens RMSE	Devens EVA
PilotNet [15]	0.00694	0.108	0.00946	0.713
Ours	0.00665	0.182	0.00849	0.763
Maqueda et al. [18]	0.00624	0.275	0.00710	0.823
Maqueda et al. with ROI cropping	0.00707	0.109	0.0113	0.577
Maqueda et al. with 1-channel integration	0.00666	0.175	0.0115	0.572
Maqueda et al. with both	0.00707	0.0907	0.0114	0.558

Table 1: Results of models in 3 on our datasets

5.2 Visualization

In order to understand the objects that determine the models' driving decisions, we employ VisualBackProp [29], which has already been used by of the authors of PilotNet to explain their model in [30]. Using this method, the salient (important) objects of two example inputs for the three models are plotted in green in Figure 4.

From the saliency maps, we could see that the salient objects in all RGB convolutional layers, both in PilotNet and in our model, reflect the way human drivers would use the visual cues on the road: the lane marks and the edges of the road are clearly highlighted. However, event-based models, especially in [18] where the image is not cropped, pay much attention to information that is insignificant to human drivers, such as the vegetation on the sides. In comparison, our event-frame layers are more understandable to human in focusing on important features like the lane marks, showing the effectiveness of our methods.

5.3 Real-World Driving and Discussions

After the models have shown good offline metric performance on the Devens dataset, they is loaded onto the NVIDIA Drive PX2 for the road test. We drove each model for about three minutes on roads that it has not seen before. To calculate the model's performance, we use the percentage autonomy metric proposed by PilotNet [15]. Assuming each intervention in real life would require a total of six seconds, this metric is calculated by counting the number of interventions, multiplying by 6 seconds, dividing by the elapsed time of the test, and then subtracting the result from 1:

$$autonomy = (1 - \frac{(number\ of\ interventions) \cdot 6\ seconds}{elapsed\ time\ [seconds]}) \cdot 100 \tag{4}$$

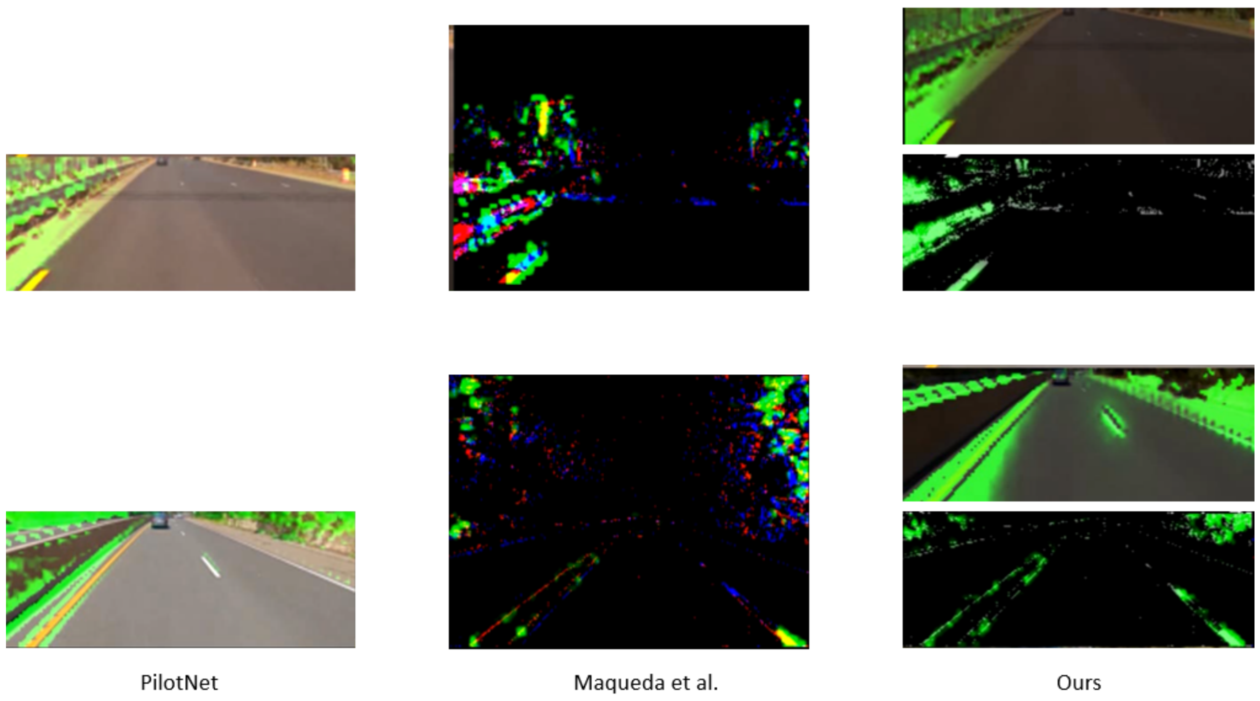


Figure 4: **Comparison of saliency between PilotNet [15], Maqueda et al. [18], and Ours**

The results are shown in Table 2.

Model	autonomy
PilotNet [15]	66%
Maqueda et al. [18]	0%
Ours	45%

Table 2: Results of models in 3 on our datasets

While we have predicted that model from [18] may not be able to drive well, it is to our surprise that it totally fails to drive for any significant distance. Moreover, despite our best efforts to prevent our model from cheating in offline datasets through ROI-cropping and single-channel integration, and despite the more human understandable salient objects in visualization, adding event-based vision actually degrades our model's performance in driving a full-scale autonomous vehicle.

Our results have revealed a fundamental challenge in event-base active-control end-to-end autonomous driving, that event-based models may need a different training scheme than that of PilotNet. While traditional RGB frames utilized by PilotNet provide information about the scene topology, the event-based cameras provide additional information about the existing motion of the car in relation to the scene, as events naturally correspond to moving edges. This allows the model to simply predict the existing motion of the car without understanding the active control from the scene topology. This challenge may go beyond event-based vision as well. For example, using a RNN-based model on RGB input may also encourages the model to identify and output the current motion of the car rather than learning the active control.

In order to combat this challenge in event-based end-to-end driving, the new training scheme needs to force the model to learn reasoning about future actions with a feedback loop that impacts the future state, which is one of our future research areas.

6 Conclusions

In this work, we designed a novel model that fuses the benefits from traditional frame-based cameras and bio-inspired event-based cameras. We demonstrate the new model's ability to perform well on datasets and focus on human understandable features. Experimental results on a full-scale autonomous vehicle show that while our models perform better than previous state-of-art event-based model, future research needs to be done for event-based models to outperform traditional frame-based models in real-world end-to-end driving.

In the future, we intend to work on improving the current training scheme to teach the models reasoning about future actions with a feedback loop that impacts the future state.

References

[1] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128$\times$ 128 120 db 15 μs latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 566–576, Feb 2008.

[2] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, *et al.*, "Event-based vision: A survey," *arXiv preprint arXiv:1904.08405*, 2019.

[3] I.-A. Lungu, F. Corradi, and T. Delbrück, "Live demonstration: Convolutional neural network driven by dynamic vision sensor playing roshambo," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–1, IEEE, 2017.

[4] X. Lagorce, G. Orchard, F. Galluppi, B. E. Shi, and R. B. Benosman, "Hots: a hierarchy of event-based time-surfaces for pattern recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 7, pp. 1346–1359, 2016.

[5] D. P. Moeys, F. Corradi, E. Kerr, P. Vance, G. Das, D. Neil, D. Kerr, and T. Delbrück, "Steering a predator robot using a mixed frame/event-driven convolutional neural network," in *2016 Second International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*, pp. 1–8, IEEE, 2016.

[6] J. Levinson and S. Thrun, "Robust vehicle localization in urban environments using probabilistic maps," in *2010 IEEE International Conference on Robotics and Automation*, pp. 4372–4378, IEEE, 2010.

[7] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial intelligence*, vol. 128, no. 1-2, pp. 99–141, 2001.

[8] A. Amini, B. Horn, and A. Edelman, "Accelerated convolutions for efficient multi-scale time to contact computation in julia," *arXiv preprint arXiv:1612.08825*, 2016.

[9] A. A. Assidiq, O. O. Khalifa, M. R. Islam, and S. Khan, "Real time lane detection for autonomous vehicles," in *2008 International Conference on Computer and Communication Engineering*, pp. 82–88, IEEE, 2008.

[10] U. Lee, S. Yoon, H. Shim, P. Vasseur, and C. Demonceaux, "Local path planning in a complex environment for self-driving car," in *The 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent*, pp. 445–450, IEEE, 2014.

[11] C. Richter, W. Vega-Brown, and N. Roy, "Bayesian learning for safe high-speed navigation in unknown environments," in *Robotics Research*, pp. 325–341, Springer, 2018.

[12] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, "Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 9, pp. 2994–3008, 2017.

[13] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Transactions on control systems technology*, vol. 15, no. 3, pp. 566–580, 2007.

[14] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, pp. 305–313, 1989.

[15] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[16] A. Amini, L. Paull, T. Balch, S. Karaman, and D. Rus, "Learning steering bounds for parallel autonomous systems," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, IEEE, 2018.

[17] S. Hecker, D. Dai, and L. Van Gool, "End-to-end learning of driving models with surround-view cameras and route planners," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 435–453, 2018.

[18] A. I. Maqueda, A. Loquercio, G. Gallego, N. García, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5419–5427, 2018.

[19] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (USA), pp. 807–814, Omnipress, 2010.

[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 448–456, JMLR.org, 2015.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, Jan. 2014.

[22] F. Naser, D. Dorhout, S. Proulx, S. D. Pendleton, H. Andersen, W. Schwarting, L. Paull, J. Alonso-Mora, M. H. Ang, S. Karaman, *et al.*, "A parallel autonomy research platform," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 933–940, IEEE, 2017.

[23] A. Amini, G. Rosman, S. Karaman, and D. Rus, "Variational end-to-end navigation and localization," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8958–8964, IEEE, 2019.

[24] "LI-AR0231-GMSL-xxxH leopard imaging inc data sheet," 2019.

[25] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240 × 180 130 db 3 μs latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, pp. 2333–2341, 2014.

[26] "NVIDIA Drive PX2 SDK documentation.," 2019.

[27] F. Codevilla, A. M. López, V. Koltun, and A. Dosovitskiy, "On offline evaluation of vision-based driving models," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 236–251, 2018.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] M. Bojarski, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, U. Muller, and K. Zieba, "Visualbackprop: efficient visualization of cnns," *arXiv preprint arXiv:1611.05418*, 2016.

[30] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller, "Explaining how a deep neural network trained with end-to-end learning steers a car," *arXiv preprint arXiv:1704.07911*, 2017.